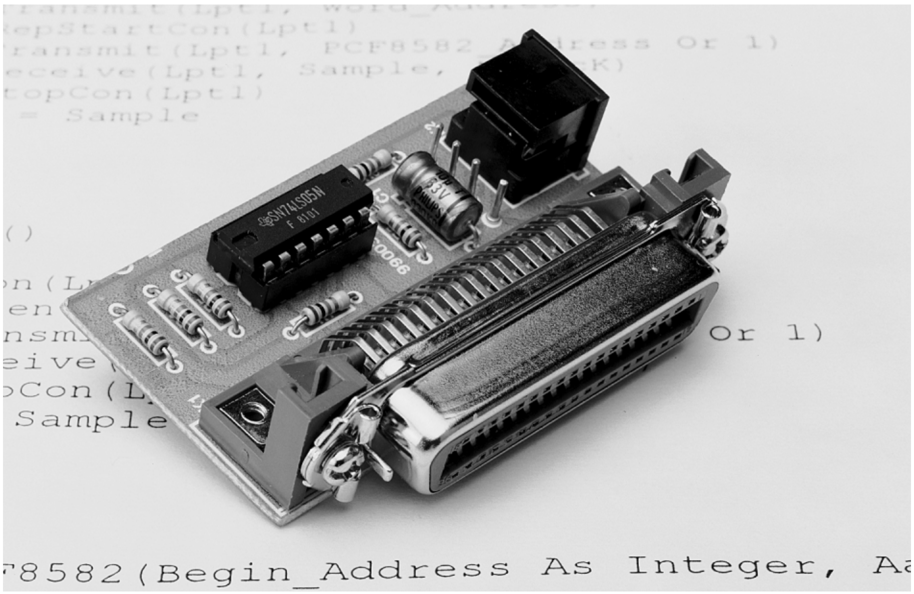
Many hobbyists use their PCs to control circuits via an I^{2}C interface. In this article, we show how the necessary programming can be made considerably easier with the help of a simple interface card and a special Visual Basic DLL. Now you can control all your I^{2}C hardware via Visual Basic under Windows.

Design by R. Van Lier

I^{2}C interface for the printer port

easily programmed with Visual Basic



I^{2}C, the Inter-IC bus, was developed by Philips to allow ICs to communicate with each other. Many modern pieces of equipment, such as tuners, television sets and video recorders utilize this interface. Nowadays, the major microcontroller families support this interface, and driving the I^{2}C bus is furthermore not all that difficult.

If the I^{2}C bus is used to control equipment via the PC, then things are quite a bit more complicated. A special hardware interface card that allows external equipment to be controlled has been described in a past issue of *Elektor Electronics*. The associated software was limited to simple device drivers at the DOS level. Presently, with practically every PC running with Windows, this approach is a bit outdated.

There's another way

Many programmers these days utilize Visual Basic, which is a graphical development environment for programming in BASIC under Windows. The most significant limitation of Visual Basic is the fact that memory locations cannot be directly addressed. Filling in the contents of hardware registers in special chips is thus out of the question. Fortunately, there is a way to work around this problem, based on using a special DLL.

The DLLs presented here contain a number of special functions, including:

- generating the start condition,
- generating a stop condition,
- sending a byte,
- receiving a byte.

There are two types of DLL available. The first is IIC.DLL, a 16-bit version that is suitable for use with Visual Basic 3 (Windows 3.1x), and the second is IIC_32.dll, a 32-bit version that is suitable for use with Visual Basic 4 and above (Windows 95 and later). These DLLs must be copied to the Windows/System folder or the Windows/System32 folder, respectively. Users of Windows NT should in principle also be able to use the 32-bit DLL, but they encounter a problem in that Windows NT does not allow the user to directly write to hardware addresses. In this case a solution is available in the form of the program TotalIO, which is distributed via the Internet. Interested parties can find it at

http://www.ise.ufl.edu/esi4161/files/direct_io/

Once this driver has been installed, it is no problem to access the I/O hardware directly from the software.

A complete concept

Support for the PCF8591 IC is already built into the DLLs. This IC contains both an A/D and a D/A converter and is used in many projects. The example program IIC.BAS (which is located on the diskette along with the drivers) shows how both the PCF8591 (A/D and D/A) and the PCF8582 (EEPROM) can be used with the interface card of this project, supported by Visual Basic. Based on these examples, you can see that using the I^{2}C interface has become very easy indeed. The only limitation with this approach is that

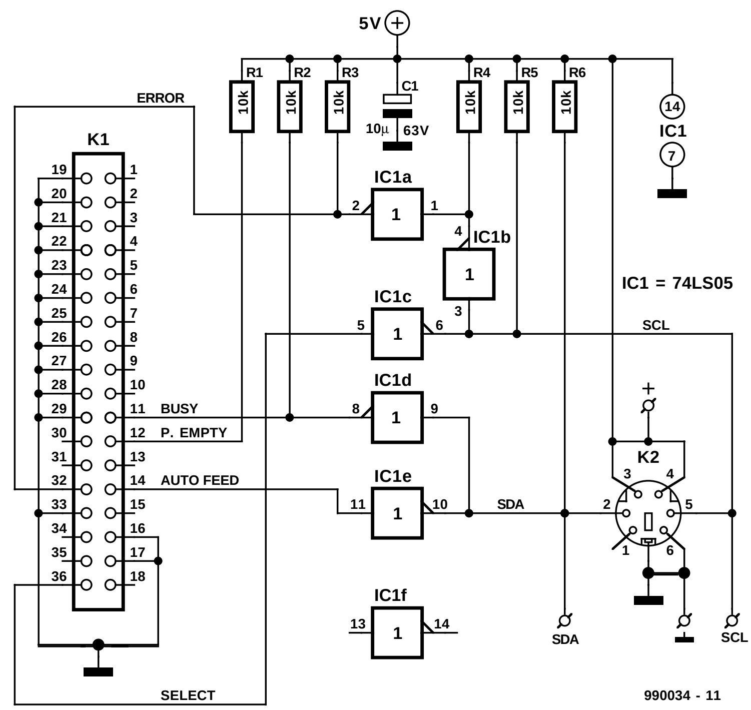


Figure 1. Schematic diagram of the I^{2}C interface, which can be connected to the PC Centronics port.

there can be only one master in the system, namely the PC.

The hardware

The DLL is based on a special I^{2}C interface that is connected to the printer port. The circuit diagram is shown in **Figure 1**. The interface is based on the standard PC Centronics port, and connects five buffers of a 74LS05 to the port. It utilizes the printer port connections Error, Busy (inputs), Paper Empty and Auto Feed (outputs). These four connections are used to generate the two I^{2}C bus control signals SCL and SDA. Both of these signals are provided via combined input/output connections, since they are both bi-directional. The two control signals and the supply voltage are routed to connector K2. The target hardware can easily be connected to the interface via a cable and a mini-DIN plug.

Regular *Elektor Electronics* readers will undoubtedly recognize the circuit, since it was already presented in the Summer Circuits issue in 1993. Only marginal changes have been made to the original circuit design.

The pull-up resistors used here are required, since the buffer and the I^{2}C driver are both open-collector devices. If R5 and R6 are the only pull-up resistors on the I^{2}C side, it is better to use a value of 3.3 kΩ.

The copper track layout and component overlay of the circuit board are shown in **Figure 2**. This board is not available through our Reader Services, so you will have to make it yourself. Fortunately, this is not difficult since the board is quite simple.

(990034-1)

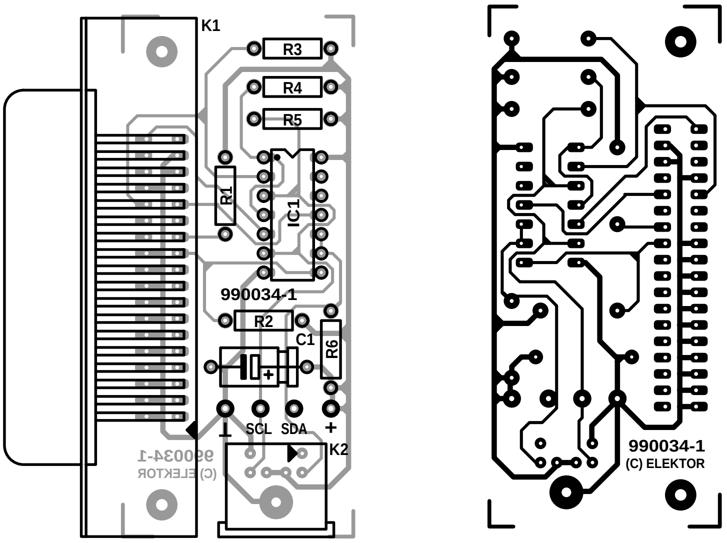
DLL functions

InitPCF8591	: initialise PCF8591
ReadPCF8591	: read byte from ADC
ReadbufPCF8591	: read n bytes from ADC
WritePCF8591	: write n bytes to DAC
StartCon	: generate start condition
RepStartCon	: generate repeated start condition
Transmit	: transmit a byte
Receive	: read a byte
TestFree	: test if bus free
TestConnected	: check if interface and supply present

Error reports

0	Error 0000	: No errors
1	Error 0001	: IIC bus not free
2	Error 0002	: No acknowledge received
3	Error 0003	: Could not generate a stop condition
4	Error 0004	: IIC interface has no power supply
5	Error 0005	: IIC interface not connected
6	Error 0006	: Undefined error occurred

Figure 2. The copper track layout and component overlay for the interface board.



COMPONENTS LIST

Resistors:
R1-R3 = 10kΩ

Capacitor:
C1 = 10µF 63V

Semiconductor:
IC1 = 74LS05

Miscellaneous:
K1 = Centronics socket (female), right-angle PCB mounting
K2 = 6-pin mini-DIN socket (female), right-angle PCB mounting
PC1-PC4 = 4 wire-wrap pins
Control software on 3.5-inch diskette (DLLs and example program), order code 996004-1.